variances specified relative to the nominal price are preferably specified in terms of a uniform measure. More preferably, the uniform measure is a monetary measurement, such as U.S. dollars or any other currency.

Please replace the paragraph beginning at page 11, line 12, with the following rewritten paragraph:

The variances to the attributes may be expressed in a number of ways and need not vary linearly relative to price. In the example shown in **FIG. 3**, the buyer specifies a nominal value of 5 for Quality 1 and the variances to Quality 1 are such that each unit of decrease in the value of Quality 1 results in an increase of 10 in price relative the nominal price of 100, up to a maximum increase of 20 for any lower value of Quality 1. In this example, Quality 1 is a negative quantity attribute, such as features size for a semiconductor chip, i.e. a lower numerical quantity represents a better product or service. Thus, if the minimum value of Quality 1 is 1, then Quality 1 values of 1, 2, or 3 all result in an increase of 20 relative to the nominal price of 100. Further, by not specifying values of Quality 1 greater than 6, the market system preferably deems that all bids having values greater than 6 of Quality 1 would not satisfy or match the set of bids represented by the input screen 300.

Please replace the paragraph beginning at page 18, line 3, with the following rewritten paragraph:

For each pair of matching buyer-seller bids, a surplus is determined in step 708. The surplus is the difference between the buyer price and the seller price for a given matching buyer-seller pair of bids, the seller price being less

than or equal to the buyer price. For each buyer-seller pair, a matching pair of bids having a highest surplus among all matching bids of the buyer-seller pair is selected in step 710.

Please replace the paragraph beginning at page 20, line 10, with the following rewritten paragraph:

As shown, the maximal weighted matching 900 includes edges 902, 904, and 906 such that the combination of the best matches between Buyer A and Seller A, Buyer B and Seller C, and Buyer D and Seller B results in a highest overall surplus. The Added Seller is not part of the maximal weighted matching 900 because all edges to the Added Seller have a weight of 0 and thus the Added Seller does not contribute to the overall surplus.

Please replace the paragraph beginning at page 19, line 15, with the following rewritten paragraph:

Where there are no matching bids between a buyer and a seller, an edge having a weight of 0 may nonetheless be added. However, if there are no matching bids between a buyer and any of the sellers (or between a seller and any of the buyers), the corresponding buyer (or seller) node may be removed from the weighted bipartite graph 800. If after the removal of all unmatched buyers and/or sellers there are an unequal number of buyers and sellers, a dummy or added buyer or seller node is preferably added to the set of nodes such that there are an equal number of buyer and seller nodes. Thus, given N number of resulting buyer nodes and N number of resulting seller nodes, the weighted bipartite graph 800 preferably includes a total of N^2 edges.

Please replace the paragraph beginning at page 20, line 15, with the following rewritten paragraph:

Determination of the maximal weighted matching of a weighted bipartite graph, also known as the assignment problem, is well known to those of ordinary skill in the art and described, in for example, Ahuja, Ravindra K., Thomas L. Magnanti, and James B. Orlin, "Network Flows: Theory, Algorithms, and Applications," 1993, (see, in particular Section 12.4). The Ahuja reference discloses the bipartite weighted matching problem; given a weighted bipartite network $G = (N_1 \cup N_2, A) with \mid N_1 \mid = \mid N_2 \mid$ and arc weights c_{ij}, finds a perfect matching of minimum weight. Here, the network *G* is directed or undirected. If the network is directed, for each arc $(i, j) \in A$, $i \in N_1$ *and* $j \in N_2$. If the network is undirected, the network is made directed by designating all arcs as pointing from the nodes in N_1 to those in N_2. Therefore, the following examples assume that *G* is a directed graph.

The assignment problem is a special case of the minimum cost flow problem and can be stated as the following linear program.

$$\text{Minimize} \sum_{\{j:(i,j)\in A\}} c_{ij}x_{ij}$$

subject to

$$\sum_{\{j:(i,j)\in A\}} x_{ij} = 1 \quad \text{for all } i \in N_1,$$

$$\sum_{\{j:(i,j)\in A\}} x_{ij} = 1 \quad \text{for all } i \in N_2,$$

$$x_{ij} \geq 0 \quad \text{for all } (i, j) \in A.$$

Since the weighted bipartite matching problem may be formulated as this special type of flow problem, it is not too surprising to learn that most algorithms for the assignment problem can be viewed as adaptations of algorithms for the minimum cost flow problem. However, the special structure of the assignment problem often permits simplification of these algorithms and to obtain improved bounds on their running times.

One popular algorithm for the assignment problem is a specialization of the network simplex algorithm and the successive shortest path algorithm and its many variants, that are well known to those of ordinary skill in the art. Examples of algorithms to solve a maximal weighted matching or assignment problem include specialization of a network simplex algorithm, successive shortest path algorithm, Hungarian algorithm, relaxation algorithm, and cost scaling algorithm for a minimum cost flow problem, as will be further described below. Many of these algorithms can be viewed as various adaptations of algorithms for a minimum cost flow problem. Any such or other suitable algorithms may be utilized to determine the maximal weighted matching of multi-attribute bids. The following briefly describe some of these successive shortest path-based algorithms and the cost scaling algorithm.

The successive shortest path algorithm, which is well known to those of ordinary skill in the art, obtains shortest path distances from a supply node to all other nodes in a residual network, uses these distances to update node potentials and then augments flow from that supply node to a demand node. The successive shortest path algorithm, when applied to the assignment problem, would augment one unit flow in every iteration, which would amount to assigning one additional node in N_1. Consequently, if $S(n, m, C)$ denotes the time needed to solve a shortest path problem with nonnegative arc lengths and

$n_1 = |N_1 1|$, the algorithm would terminate within n_1 iterations and would require $O(n_1 S(n, m, C))$ time.

The Hungarian algorithm, which is well known to those of ordinary skill in the art, is a direct implementation of the primal-dual algorithm for the minimum cost flow problem. The primal-dual algorithm first transforms the minimum cost flow problem into a problem with a single supply node s^* and a single demand node t^*. At every iteration, the primal-dual algorithm computes shortest path distances from s^* to all other nodes, updates node potentials, and then solves a maximum flow problem that sends the maximum possible flow from node s^* to node t^* over arcs with zero reduced costs. When applied to the assignment problem, this algorithm terminates within n_1 iterations since each iteration sends at least one unit of flow, and hence assigns at least one additional node in N_1. The time required to solve shortest path problems in all these iterations is $O(n_1 S(n, m, C))$. Next consider the total time required to establish maximum flows. The labeling algorithm for solving the maximum flow problem would require a total of $O(nm)$ time because it would perform n augmentations and each augmentation requires $O(m)$ time. The dominant portion of these computations is the time required to solve shortest path problems. Consequently, the overall running time of the algorithm is $O(n_1 S(n, m, C))$.

The relaxation algorithm, which is closely related to the successive shortest path algorithm, is also well known to those of ordinary skill in the art and is another popular approach for solving the assignment problem. This algorithm relaxes the constraint $\sum_{\{j:(i,j)\in A\}} x_{ij} = 1$ for all $i \in N_2$, thus allowing any node in N_2 to be assigned to more than one node in N_1. To solve the relaxed problem: Assign each node $i \in N_1$ to any node $j \in N_2$ with the minimum cost c_{ij} among all arcs in $A(i)$. As a result, some nodes in N_2 might be unassigned while some other nodes are over assigned (i.e., assigned to more than one node in N_1). The algorithm

then gradually converts this solution to a feasible assignment while always maintaining the reduced cost optimality condition. At each iteration the algorithm selects an over-assigned node k in N_2, obtains shortest path distances from node k to all other nodes in a residual network with reduced costs as arc lengths, updates node potentials, and augments a unit flow from node k to an unassigned node in N_2 along the shortest path. Since each iteration assigns one more node in N_2 and never converts any assigned node into an unassigned node, within n_1 such iterations, the algorithm obtains a feasible assignment. The relaxation algorithm maintains optimality conditions throughout. Therefore, the shortest path problems have nonnegative arc lengths, and the overall running time of the algorithm is $O(n_1 S(n, m, C))$.

The cost scaling algorithm, which is well known to those of ordinary skill in the art, is an adaptation of the cost scaling algorithm for the minimum cost flow problem. The cost scaling algorithm performs $O(nC))$ scaling phases and the generic implementation requires $O(n^2 m)$ time for each scaling phase. The bottleneck operation in each scaling phase is performing nonsaturating pushes which require $O(n^2 m)$ time; all other operations, such as finding admissible arcs and performing saturating pushes, require $O(nm)$ time. When applying the cost scaling algorithm to the assignment problem, each push is a saturating push since each arc capacity is one. Consequently, the cost scaling algorithm solves the assignment problem in $O(nm \log(nC))$ time.

A modified version of the cost scaling algorithm has an improved running time of $O(\sqrt{n}\, m \log(nC))$, which is the best available time bound for assignment problems satisfying the similarity assumption. This improvement rests on decomposing the computations in each scaling phase into two subphases. In the first subphase, apply the usual cost scaling algorithm is applied with the difference that whenever a node is relabeled more than $2\sqrt{n}$ times, this node is

set aside and not examined further. When all (remaining) active nodes have been set aside, the second subphase is initiated. It is possible to show that the first subphase requires $O(\sqrt{n_1}m)$ time, and when it ends, the network will contain at most $O(\sqrt{n_1})$ active nodes. The second subphase makes these active nodes inactive by identifying "appropriate shortest paths" from nodes with excesses to nodes with deficits and augmenting unit flow along these paths. The algorithm uses Dial's algorithm to identify each such path in $O(m)$ time. Consequently, the second subphase also runs in $O(\sqrt{n_1}m\log(nC))$.

In the Abstract

A system and a method for matching multi-attribute auction bids are disclosed. A set of multi-attribute bids or bid values are collected from one or more buyers and one or more sellers. The set of bid values may include variances from nominal bid values. Buyer and seller bids are generated from the set of buyer and seller multi-attribute bid values after predetermined attribute values, if any, are added. A pair of bids between each buyer and each seller having a highest surplus is selected. The method generates a weighted bipartite graph having buyer nodes and seller nodes and an edge between each buyer node and each seller node, each edge having the highest surplus of the pair of bids between the buyer and seller as a weight. The maximal weighted matching bids from the highest surplus pairs of bids are determined using the weighted bipartite graph.